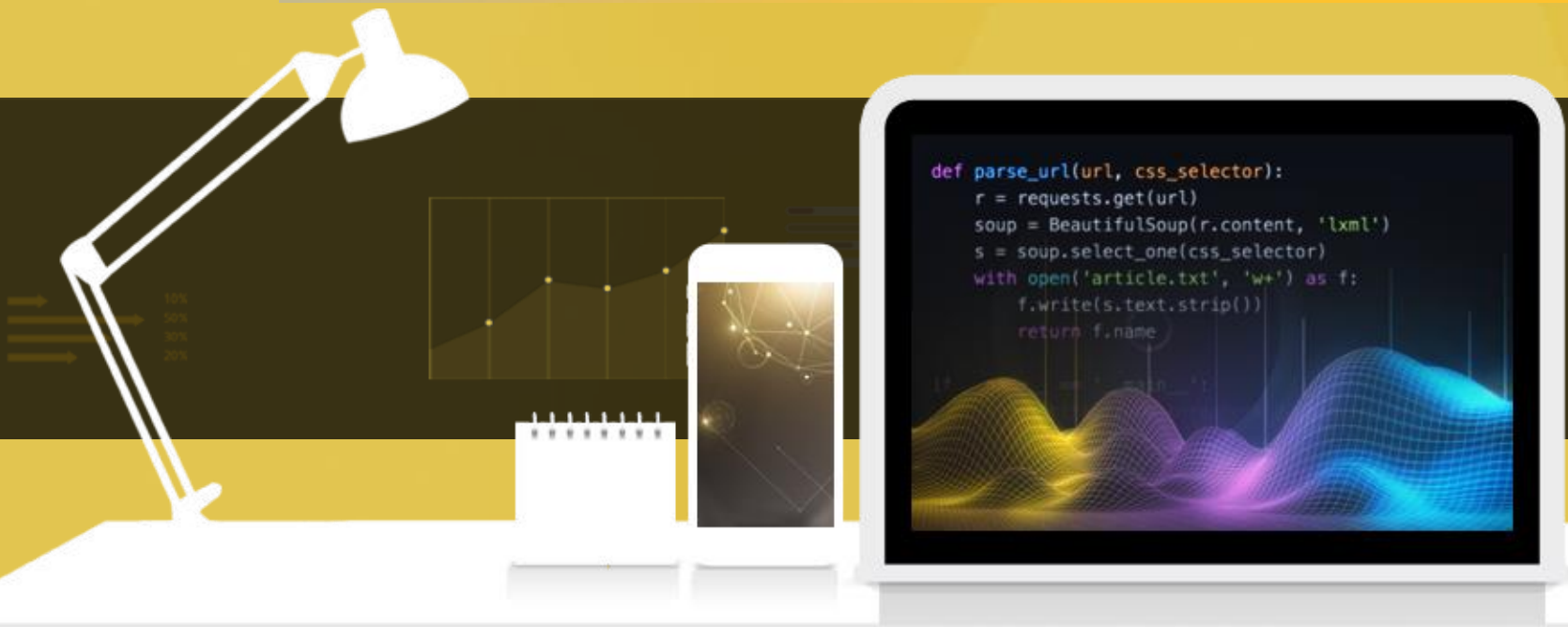


맷플롯립 (matplotlib)



소프트웨어융합대학원
진혜진

목차



맷플롯립



01 맷플롯립

- 데이터 시각화(data visualization) : 데이터 분석 결과를 쉽게 이해할 수 있도록 시각적으로 표현하고 전달
- 맷플롯립, 시본, 플롯리 외에도 bokeh, pygal, ggplot 등

1. 맷플롯립의 구조

- 맷플롯립(matplotlib) : 매트랩(matlab) 기능을 파이썬에서 그대로 사용하도록 하는 시각화 모듈
 - 엑셀의 정형화된 차트나 그래프 작성, 다양한 함수 지원
 - 매트랩을 포장(wrapping)해서 맷플롯립을 지원

```
import matplotlib.pyplot as plt
```

01 맷플롯립

1.1 파이플롯

- 맷플롯립을 이용할 때 가장 기본이 되는 객체
- 파이플롯(pyplot) 위에 그림.figure) 객체를 올리고 그 위에 그래프에 해당하는 축(axes)을 올림
- 그림 위에 축을 여러 장 올리면 여러 개의 그래프 작성

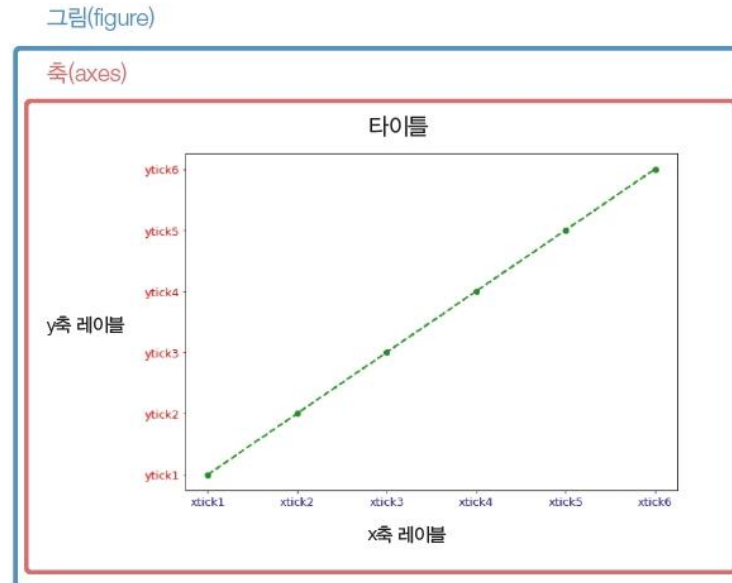
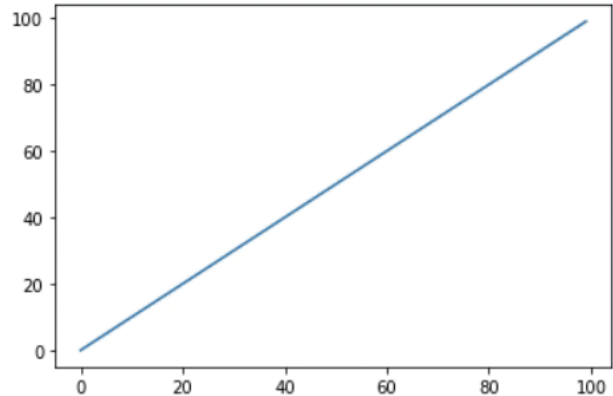


그림 5-1 파이플롯(pyplot), 그림(figure), 축(axes)의 개념

01 맷플롯립

In [1]:	<pre>import matplotlib.pyplot as plt # matplotlib 모듈 호출 X = range(100) Y = range(100) plt.plot(X, Y)</pre>
Out [1]:	

- X 객체와 Y 객체 값 쌍으로 좌표평면 위에 점을 찍음
- plot 함수로 점들을 연결

01 맷플롯립

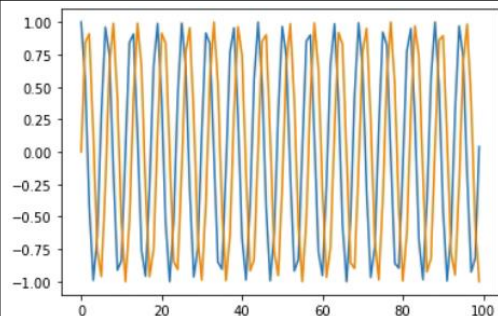
```
In [2]: import numpy as np # numpy 모듈 호출

X_1 = range(100)
Y_1 = [np.cos(value) for value in X]

X_2 = range(100)
Y_2 = [np.sin(value) for value in X]

plt.plot(X_1, Y_1)
plt.plot(X_2, Y_2)
plt.show()
```

Out [2]:



- pyplot 객체 내부에 있는 하나의 그림 객체 위에 코사인 그래프와 사인 그래프를 그림

01 맷플롯립

1.2 그림과 축

- 그림은 그래프를 작성하는 밑바탕이 됨
- 축은 실제로 그래프를 작성하는 공간

```
In [3]: fig, ax = plt.subplots() # (1) figure와 axes 객체 할당

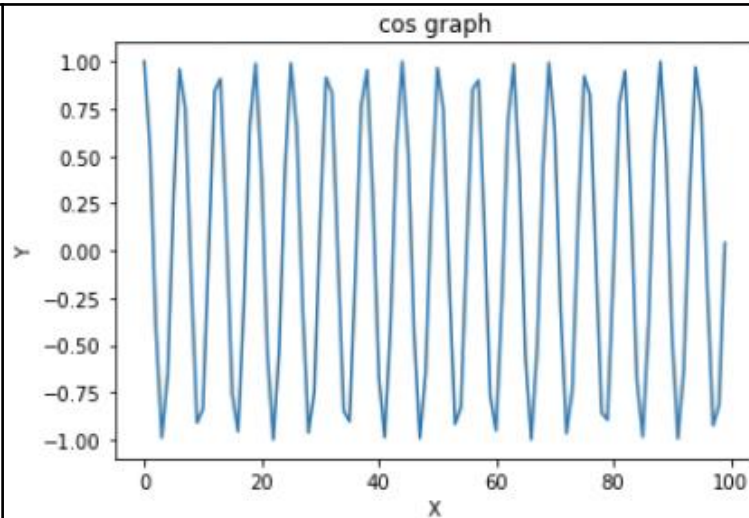
X_1 = range(100)
Y_1 = [np.cos(value)
        for value in X]

ax.plot(X_1, Y_1) # (2) plot 함수를 사용하여 그래프 생성
ax.set(title='cos graph', # (3) 그래프 제목, X축 라벨, Y축 라벨 설정
        xlabel='X',
        ylabel='Y');

plt.show() # (4) 그래프 출력
```


01 맷플롯립

Out [3]:

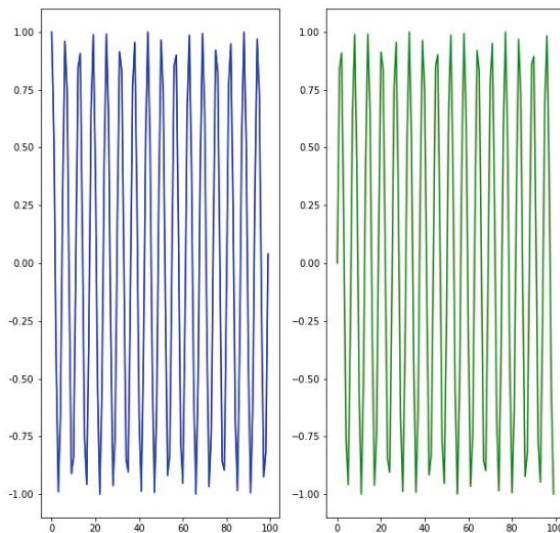


01 맷플롯립

In [4]:

```
fig = plt.figure()      # (1) figure 반환  
fig.set_size_inches(10,10) # (2) figure의 크기 지정  
  
ax_1 = fig.add_subplot(1,2,1) # (3) 첫 번째 그래프 생성  
ax_2 = fig.add_subplot(1,2,2) # (4) 두 번째 그래프 생성  
  
ax_1.plot(X_1, Y_1, c="b") # (5) 첫 번째 그래프 설정  
ax_2.plot(X_2, Y_2, c="g") # (6) 두 번째 그래프 설정  
plt.show()              # (7) 그래프 출력
```

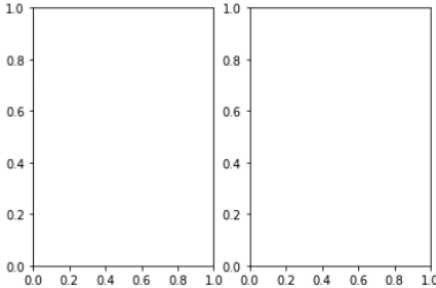
Out [4]:



01 맷플롯립

1.3 서브플롯 행렬

- 축을 여러 개 만들 때 서브플롯으로 축 객체 공간 확보
 - 그림 객체에서 `add_subplot` 함수 사용
 - 또는 `plot` 객체에서 `subplots` 함수 사용

In [5]:	<pre>fig, ax = plt.subplots(nrows=1, ncols=2) print(ax)</pre>
Out [5]:	
In [6]:	<pre>print(type(ax))</pre>
Out [6]:	<pre><class 'numpy.ndarray'></pre>

- `ax` 변수에 축 객체가 넘파이 배열 타입으로 생성됨

01 맷플롯립

```
In [7]: import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-1,1,100) # (1) x 값과 y_n 값 생성
y_1 = np.sin(x)
y_2 = np.cos(x)
y_3 = np.tan(x)
y_4 = np.exp(x)

fig, ax = plt.subplots(2, 2) # (2) 2x2 figure 객체를 생성

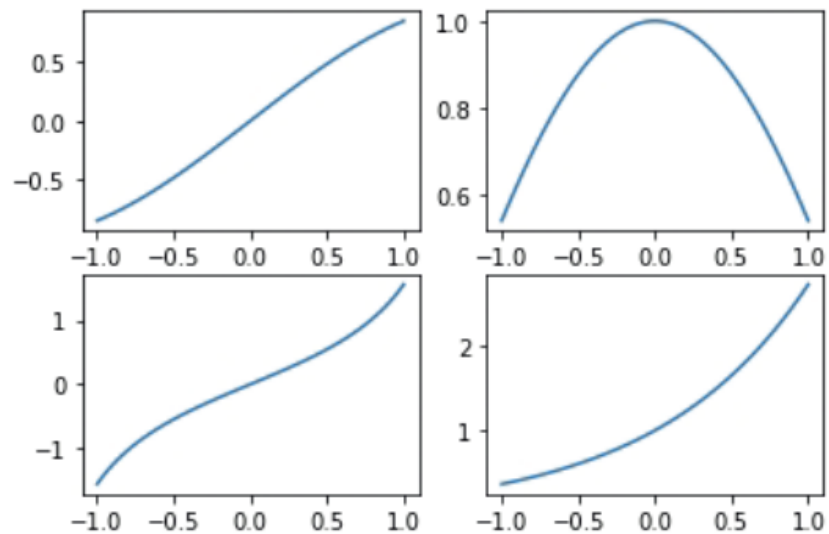
ax[0, 0].plot(x, y_1) # (3) 첫 번째 그래프 생성
ax[0, 1].plot(x, y_2) # (4) 두 번째 그래프 생성
ax[1, 0].plot(x, y_3) # (5) 세 번째 그래프 생성
ax[1, 1].plot(x, y_4) # (6) 네 번째 그래프 생성

plt.show()
```

- # (2) subplots 함수에서 2x2 행렬 그림 객체가 생성되어 ax 변수에 4개의 축 객체가 2x2 넘파이 배열 형태로 들어가 있음
- 넘파이 배열의 인덱스로 각 축 객체에 접근하여 그래프를 생성

01 맷플롯립

Out [7]:



01 맷플롯립

- 행과 열을 지정하고, 세 번째 숫자는 축의 위치

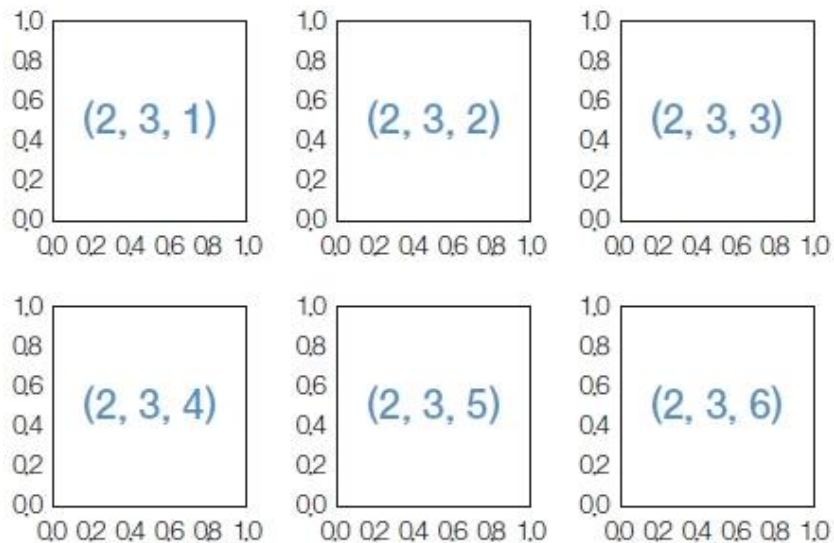
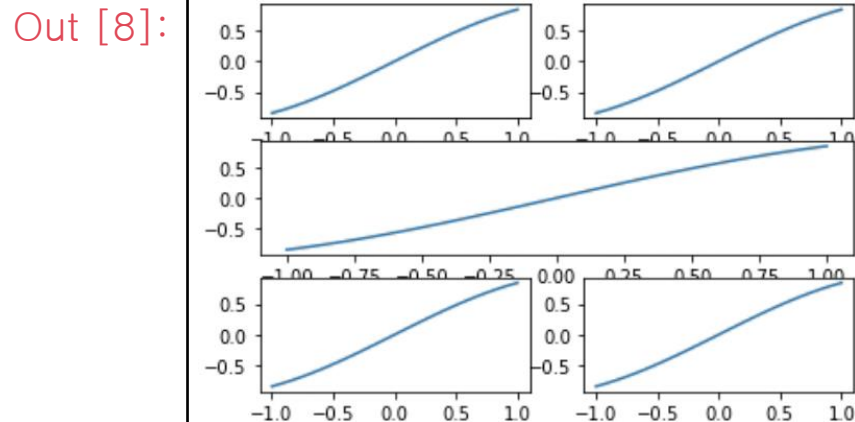


그림 5-2 지정된 행렬에서 축(axes)의 위치 배열

01 맷플롯립

```
In [8]: ax1 = plt.subplot(321) # (1) 첫 번째 공간에 axes 생성  
plt.plot(x, y_1)  
ax2 = plt.subplot(322) # (2) 두 번째 공간에 axes 생성  
plt.plot(x, y_1)  
ax3 = plt.subplot(312) # (3) 두 번째 공간에 axes 생성  
plt.plot(x, y_1)  
ax4 = plt.subplot(325) # (4) 다섯 번째 공간에 axes 생성  
plt.plot(x, y_1)  
ax5 = plt.subplot(326) # (5) 여섯 번째 공간에 axes 생성  
plt.plot(x, y_1)  
  
plt.show()import num
```



01 맷플롯립

2. 맷플롯립으로 그래프 꾸미기

2.1 색상

- color 또는 c 매개변수로 색상 변경
 - RGB 값을 사용해서 #을 붙여 16진법으로 색상 표현
 - 또는 b, g, r, c, m, y, k, w 등 약어 입력

01 맷플롯립

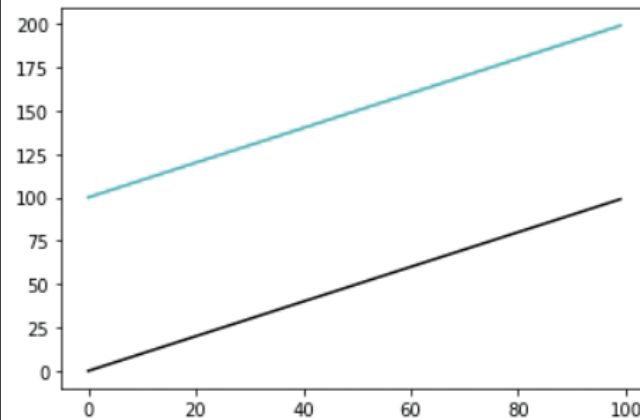
```
In [9]: X_1 = range(100)
        Y_1 = [value for value in X]

        X_2 = range(100)
        Y_2 = [value + 100 for value in X]

        plt.plot(X_1, Y_1, color="#000000")
        plt.plot(X_2, Y_2, c="c")

        plt.show()
```

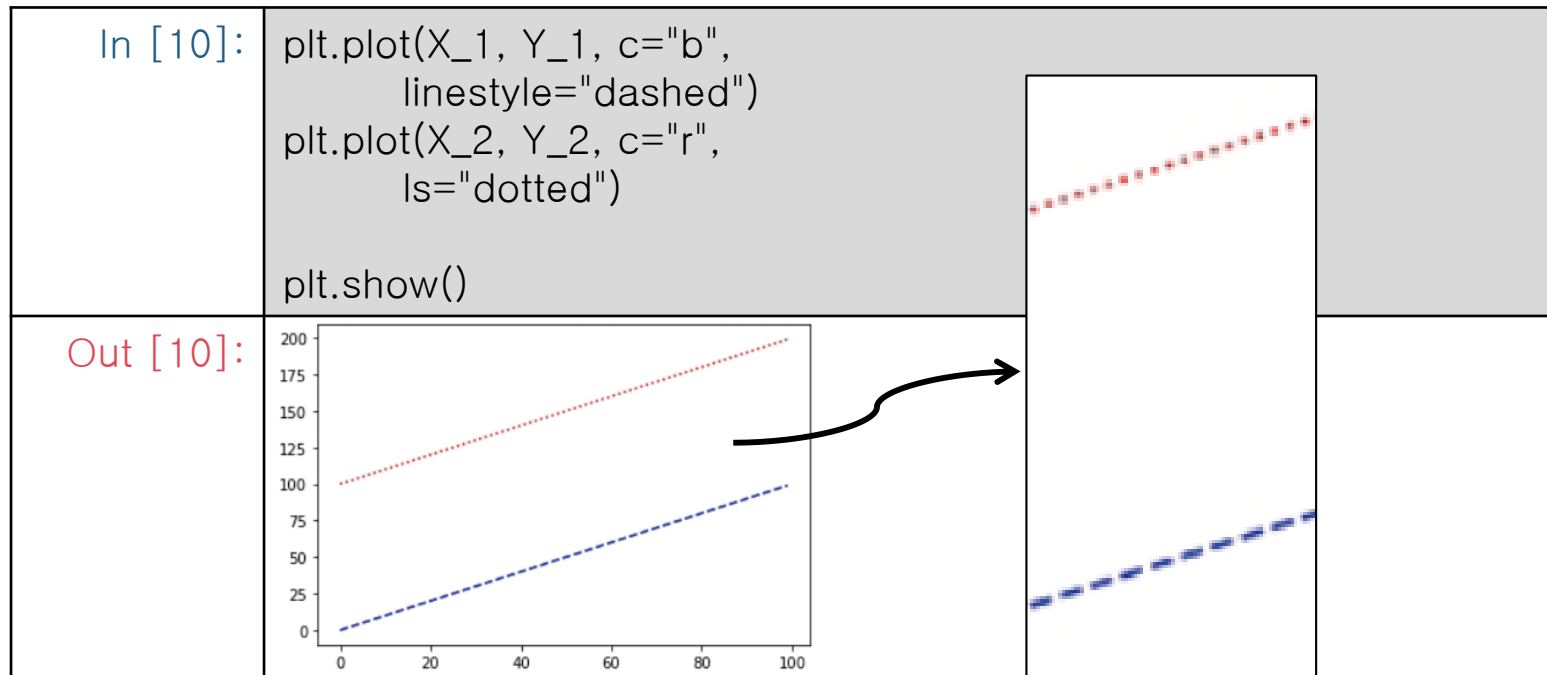
Out [9]:



01 맷플롯립

2.2 선의 형태

- linestyle 또는 ls로 선의 형태를 정의
 - dashed : 점선 형태 solid : 실선 형태



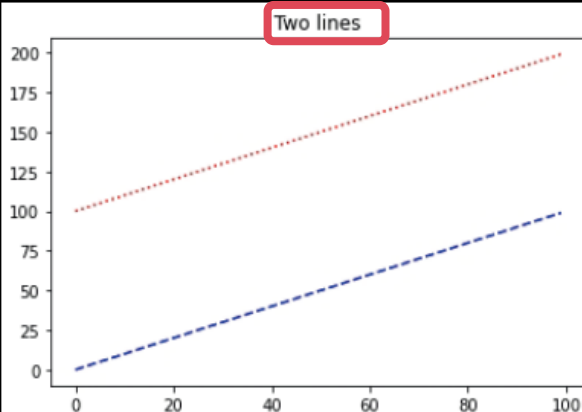
01 맷플롯립

2.3 제목

- 축 객체마다 제목을 달 수 있음

```
In [11]: plt.plot(X_1, Y_1, c="b",  
                 linestyle="dashed")  
plt.plot(X_2, Y_2, c="r",  
         ls="dotted")  
  
plt.title("Two lines")  
plt.show()
```

Out [11]:



01 맷플롯립

```
In [12]: fig = plt.figure()
fig.set_size_inches(10,10)

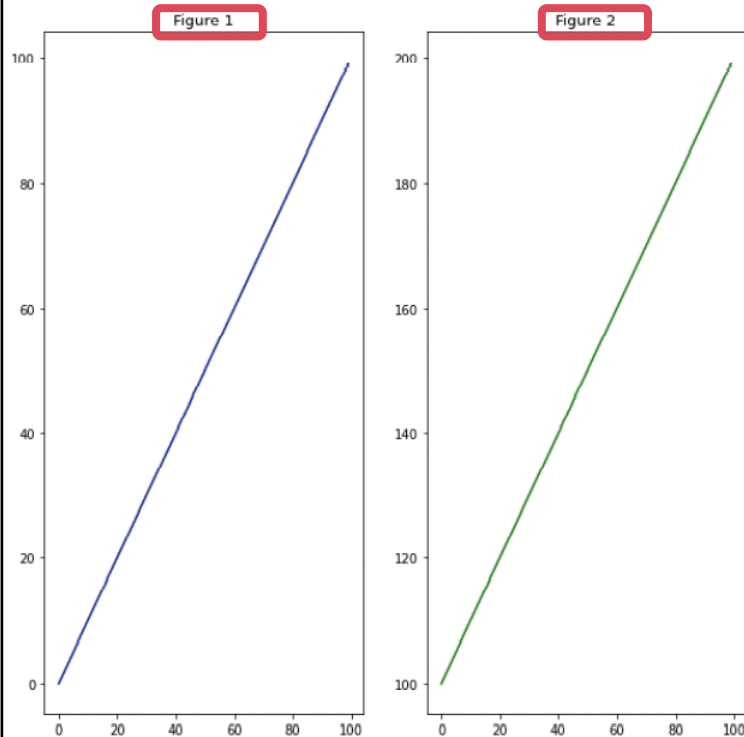
ax_1 = fig.add_subplot(1,2,1)
ax_2 = fig.add_subplot(1,2,2)

ax_1.plot(X_1, Y_1, c="b")
ax_1.set_title("Figure 1")
ax_2.plot(X_2, Y_2, c="g")
ax_2.set_title("Figure 2")

plt.show()
```

01 맷플롯립

Out [12]:



01 맷플롯립

2.4 범례

- 축 객체마다 범례를 설정할 수 있음
- legend 함수 사용하여 생성
 - shadow 매개변수로 범례에 그림자 효과 추가
 - loc 매개변수로 범례의 위치 지정
 - 값은 center, upper right 등 총 11가지
 - best라고 지정하면 적절한 위치에 범례가 놓임

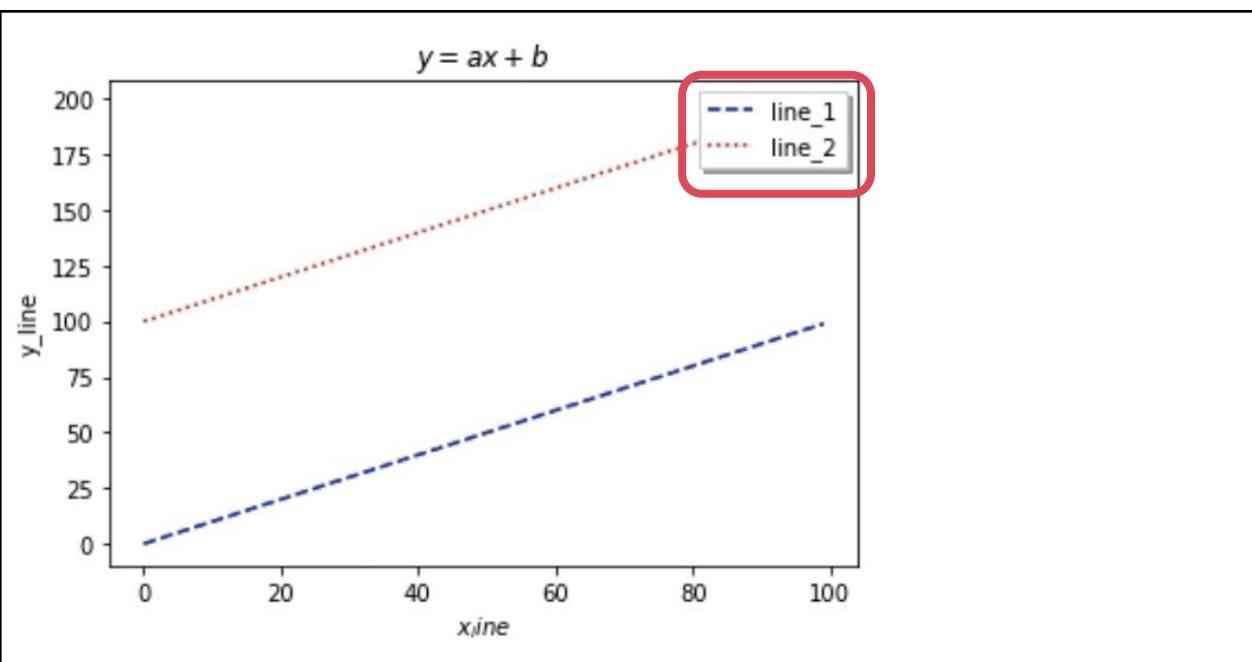
01 맷플롯립

```
In [13]: plt.plot(X_1, Y_1,
                 color="b",
                 linestyle="dashed",
                 label='line_1')
plt.plot(X_2, Y_2,
         color="r",
         linestyle="dotted",
         label='line_2')
plt.legend(
    shadow=True,
    fancybox=False,
    loc="upper right")

plt.title('$y = ax+b$')
plt.xlabel('$x_{line}$')
plt.ylabel('y_line')
```

01 맷플롯립

Out [13]:



01 맷플롯립

3. 맷플롯립에서 사용하는 그래프

3.1 산점도

- 산점도(scatter plot) : 데이터 분포를 2차원 평면에 표현
 - 매개변수 `c`는 포인트 색상을 지정
 - `marker`는 포인트 모양을 지정
 - `size`는 포인트 크기를 지정
 - `alpha`는 포인트 불투명도를 지정

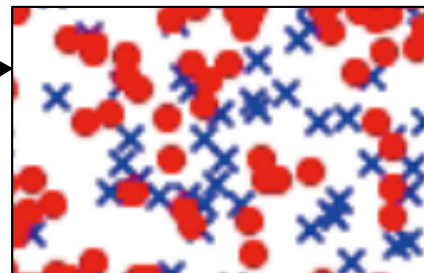
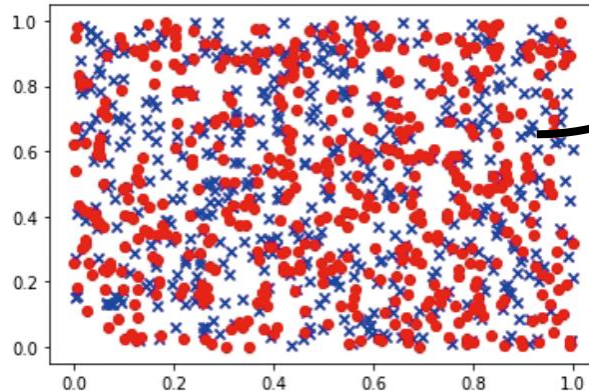
01 맷플롯립

```
In [14]: data_1 = np.random.rand(512, 2)
data_2 = np.random.rand(512, 2)

plt.scatter(data_1[:,0],
            data_1[:,1],
            c="b", marker="x")
plt.scatter(data_2[:,0],
            data_2[:,1],
            c="r", marker="o")

plt.show()
```

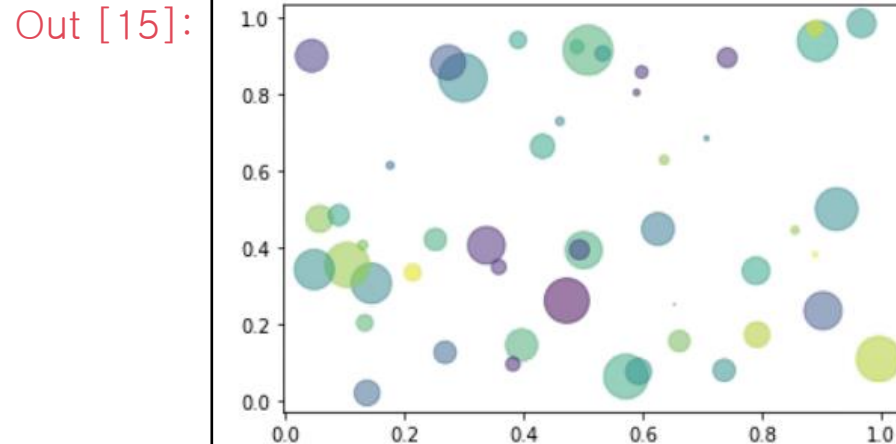
Out [14]:



01 맷플롯립

```
In [15]: N = 50
x = np.random.rand(N)
y = np.random.rand(N)
colors = np.random.rand(N)
area = np.pi * (
    15 * np.random.rand(N))**2

plt.scatter(x, y,
            s=area, c=colors,
            alpha=0.5)
plt.show()
```



01 맷플롯립

3.2 막대그래프

- 막대그래프(bar graph) : 데이터의 개수나 크기를 비교

```
In [16]: # (1) 데이터 생성
data = [[5., 25., 50., 20.],
        [4., 23., 51., 17],
        [6., 22., 52., 19]]

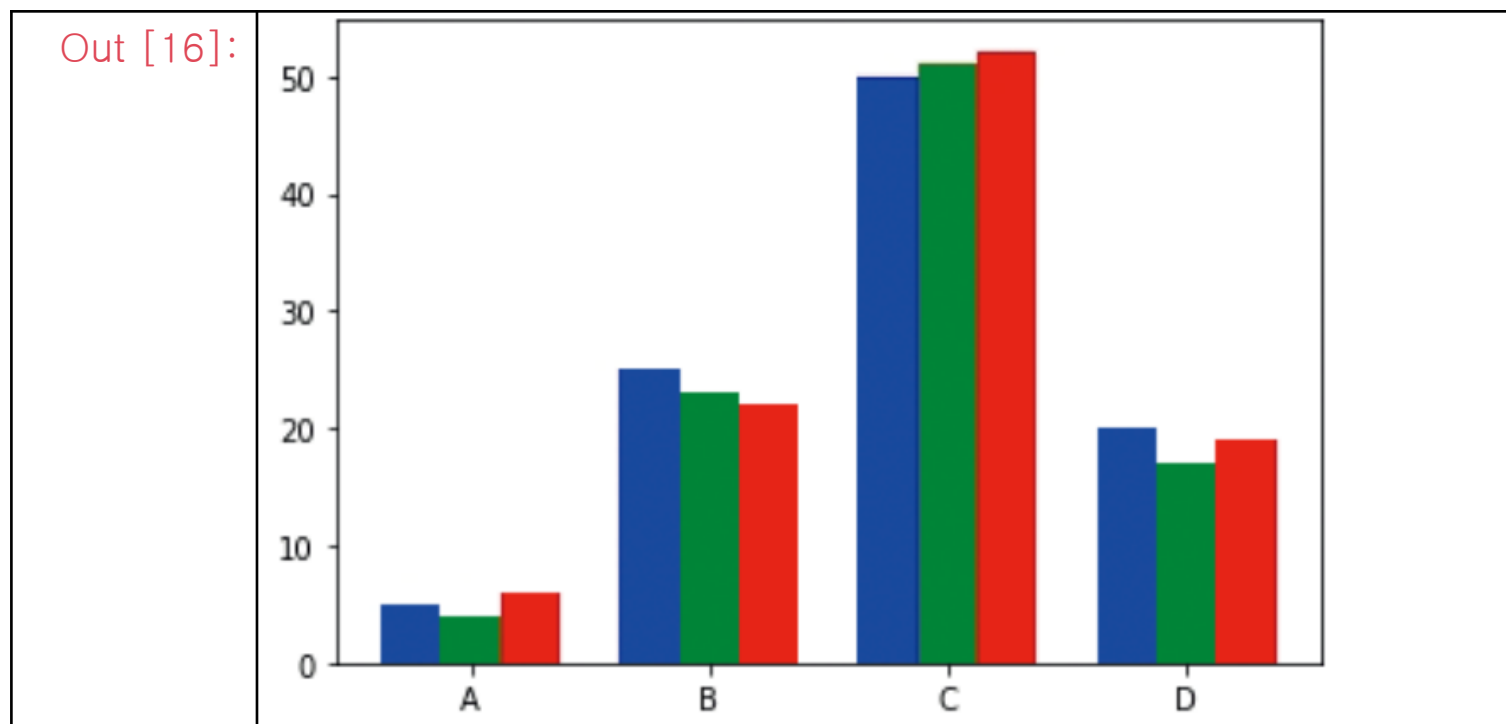
# (2) X 좌표 시작점
X = np.arange(0,8,2)

# (3) 3개의 막대그래프 생성
plt.bar(X + 0.00, data[0], color = 'b', width = 0.50)
plt.bar(X + 0.50, data[1], color = 'g', width = 0.50)
plt.bar(X + 1.0, data[2], color = 'r', width = 0.50)

# (4) X축에 표시될 이름과 위치 설정
plt.xticks(X+0.50, ("A","B","C", "D"))

# (5) 막대그래프 출력
plt.show()
```

01 맷플롯립



01 맷플롯립

3.3 누적 막대그래프

- 누적 막대그래프(stacked bar graph) :
데이터를 밑에서부터 쌓아올려 데이터를 표현

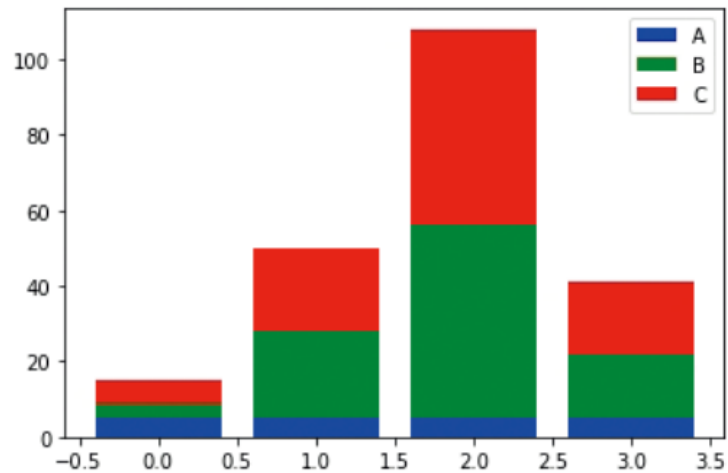
```
In [17]: data = np.array([[5., 25., 50., 20.],  
                        [4., 23., 51., 17],  
                        [6., 22., 52., 19]])  
  
color_list = ['b', 'g', 'r']  
data_label = ["A","B","C"]  
X = np.arange(data.shape[1])  
  
data = np.array([[5., 5., 5., 5.],  
                [4., 23., 51., 17],  
                [6., 22., 52., 19]])
```

01 맷플롯립

```
for i in range(3):  
    plt.bar(X, data[i],  
            bottom = np.sum(  
                data[:i], axis=0),  
            color = color_list[i],  
            label=data_label[i])
```

```
plt.legend()  
plt.show()
```

Out [17]:

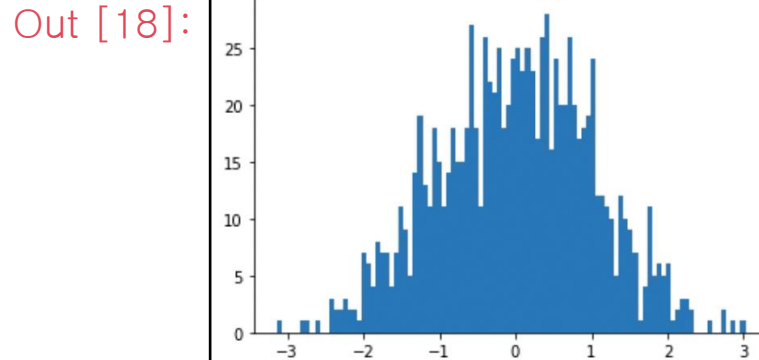


01 맷플롯립

3.4 히스토그램

- 히스토그램(histogram) : 데이터의 분포를 표현
 - hist 함수로 히스토그램 생성, 매개변수 bins로 막대 개수 지정

```
In [18]: N = 1000  
X = np.random.normal(size=N)  
  
plt.hist(X, bins=100)  
plt.show()
```



01 맷플롯립

3.4 상자그림

- 상자그림(boxplot) : 사분위수를 시각화하여 데이터의 분포와 밀집 정도를 표현

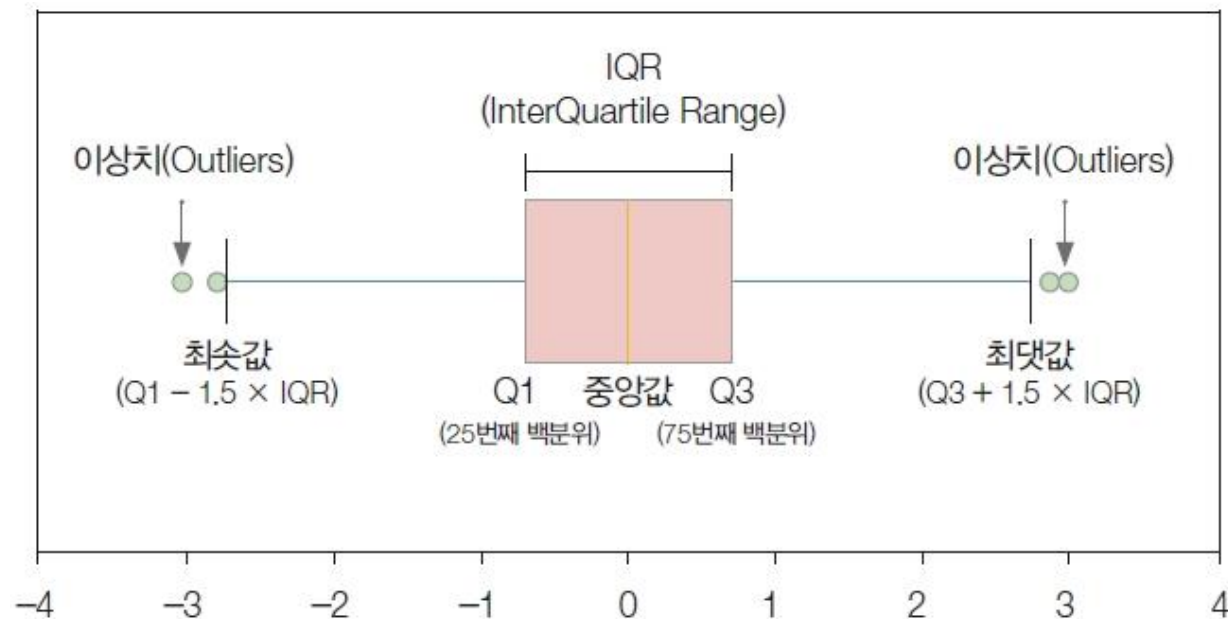


그림 5-3 상자그림의 구성 요소

01 맷플롯립

- 데이터를 작은 데이터부터 큰 데이터까지 정렬
- Q1(25%)부터 Q3(75%)까지 박스 형태로 위치시킴
- IQR(InterQuatile Range) : Q1 - Q3
- $Q1 - 1.5 \times IQR$ 을 하단 값으로, $Q3 + 1.5 \times IQR$ 을 상단 값으로
- 이상치(outlier) : 상단 값과 하단 값을 넘어가는 값들

